Neural Networks 88 (2017) 105–113

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Object class segmentation of RGB-D video using recurrent convolutional neural networks

Mircea Serban Pavel, Hannes Schulz*, Sven Behnke

Universität Bonn, Computer Science Institute VI, Friedrich-Ebert-Allee 144, 53113 Bonn, Germany

ARTICLE INFO

Article history:
Received 7 January 2016
Received in revised form 13 January 2017
Accepted 16 January 2017
Available online 30 January 2017

Keywords:
Recurrent neural networks
Computer vision
Object class-segmentation

ABSTRACT

Object class segmentation is a computer vision task which requires labeling each pixel of an image with the class of the object it belongs to. Deep convolutional neural networks (DNN) are able to learn and take advantage of local spatial correlations required for this task. They are, however, restricted by their small, fixed-sized filters, which limits their ability to learn long-range dependencies. Recurrent Neural Networks (RNN), on the other hand, do not suffer from this restriction. Their iterative interpretation allows them to model long-range dependencies by propagating activity. This property is especially useful when labeling video sequences, where both spatial and temporal long-range dependencies occur. In this work, a novel RNN architecture for object class segmentation is presented. We investigate several ways to train such a network. We evaluate our models on the challenging NYU Depth v2 dataset for object class segmentation and obtain competitive results.

1. Introduction

Current deep neural network architectures achieve superior performance on a number of computer vision tasks, such as image classification, object detection, and object class segmentation. Most of these tasks focus on extracting information from a single image. Deep neural networks compute increasingly abstract features, which simultaneously become more and more semantically meaningful, and incorporate larger contexts.

A real-world vision system will have to deal with the time dimension as well. Content is increasingly generated in the form of videos by Internet users, surveillance cameras, cars, or mobile robots. Video information can be helpful, as looking at a whole sequence instead of single frames may enable the interpretation of ambiguous measurements.

Similar to increasingly abstract features on images, we are interested in neural networks which produce high-level features on sequences. In a recursive computation, these high-level features should help to interpret the next frame in a sequence. In addition to a semantically meaningful localized content description, such features should form high-level descriptions of motions with increasing temporal context.

In this paper, we introduce a recurrent convolutional neural network architecture which produces high-level localized sequence features. We evaluate it on the NYU Depth v2 (NYUD) dataset, an RGB-D object class segmentation task, where every pixel of the input image must be labeled with the category of the object it belongs to. In this challenging and established benchmark, most methods focus on prediction based on single frames, while our method profits from image sequences.

In short, our contributions are as follows:

- We introduce a recurrent convolutional neural network model for processing image sequences.
- On toy datasets, we show that our recurrent models are able to keep an abstract state over time, track and interpret motion, and retain uncertainty.
- We show that our model improves RGB-D object class segmentation accuracy on the challenging NYUD dataset when compared to other CNN models. When combined with a CRF, RNN performance is close to carefully tuned transfer-learning approaches initialized on much larger datasets.
- We analyze the obtained networks and show that the improved performance of our network stems from the recurrent processing combined with the exploitation of temporal, i.e. video, information.

The remainder of this paper is organized as follows. Section 2 introduces our architecture and learning methods. In Section 3, we discuss the related work. We evaluate our model in Section 4, and discuss the results in Section 5.

* Corresponding author.
E-mail addresses: pavel@cs.uni-bonn.de (M.S. Pavel), schulzh@ais.uni-bonn.de (H. Schulz), behnke@cs.uni-bonn.de (S. Behnke).

http://dx.doi.org/10.1016/j.neunet.2017.01.003
0893-6080/© 2017 Elsevier Ltd. All rights reserved.

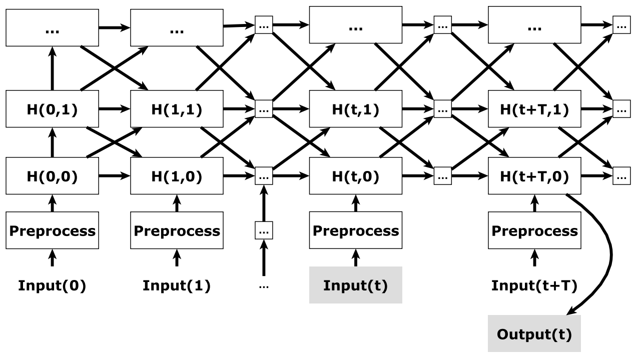


Fig. 1. Architecture of our RNN. The layers are connected to each other with valid convolutions. Upward (*forward*) connections additionally include spatial max-pooling operations, while downward (*backward*) connections include a spatial upsampling. Delay T is number of time frames between an input and corresponding output frame.

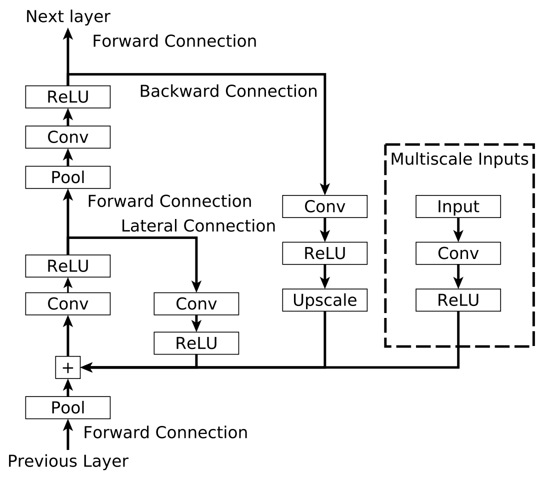


Fig. 2. Recurrent connections as viewed from a single hidden layer. Activations of a hidden layer are the result of forward connections from below, backward connections from above, and lateral connections from the same layer at previous time steps. Inputs may be provided at all scales.

2. Recurrent convolutional neural networks for image labeling

In this section, we present our network architecture. It is inspired by the Neural Abstraction Pyramid of Behnke (2003), a hierarchical recurrent convolutional neural network architecture for image processing which extends feed-forward convolutional neural networks (CNN, LeCun, Bottou, Bengio, & Haffner, 1998). As CNNs, it retains the topological image structure and ensures that features are localized. A schematic overview is shown in Figs. 1 and 2. Our base configuration (without the ability to process sequences) is a fairly standard small CNN with $L = 3$ convolutional layers, ReLU non-linearities, and interleaved spatial max-pooling.

In contrast to the Neural Abstraction Pyramid of Behnke (2003), which doubles the number of filters as the resolution of the representation is halved, we keep the number of filters to a constant 32 on all layer of the network. Initial experiments on our dataset have shown that doubling the number of filters leads to a too complex model, which takes longer to learn and is prone to overfitting.

2.1. Connection types

To process sequences, we replicate our model for T time steps and introduce connections between the temporal copies. Three types of connections exist: forward, lateral, and backward. Computationally, all connections are valid convolution operations followed by a half-rectifying point-wise non-linearity $f(x) = \max(0, x)$ (ReLU).

A hidden layer $H(t, l)$, at time step t and abstraction level l, is connected to layer $H(t + 1, l + 1)$ using a forward connection. These connections allow the vertical flow of information from the bottom of the network to the top and thus the construction of high-level features based on the low-level features. The non-linearity of the forward connections are followed by a spatial 2×2 maximum pooling operation with stride 2.

Lateral projections connect layers $H(t, l)$ and $H(t + 1, l)$. These horizontal connections can incorporate immediate spatial context from the same activation level. The intermediate context is limited by the receptive field size of the convolution filters.

Backward projections connect layer $H(t, l)$ to layer $H(t + 1, l - 1)$, and can be interpreted as providing a high-level prior for the lower, more detailed layers. Since higher layers have a coarser spatial resolution, they also provide a convenient shortcut for information that needs to travel long distances. Backward connections are immediately followed by a factor two spatial upsampling operation (i.e. every pixel is replicated four times).

Due to padded convolutions and the opposing pooling and upsampling operations, all connections coinciding on a given hidden layer have the same size, and are simply summed elementwise.

We use a convolutional feature extraction layer between the inputs and the RNN, whose weights are initialized using symmetry-k-means (Konda, Memisevic, & Michalski, 2013). Since this unsupervised initialization prevents us from controlling the gain, we ensured that these weights are not part of a feedback loop. For similar reasons, the weights of the first forward pass are excluded from weight sharing as well.

All other hidden-to-hidden connections use temporal weight sharing, i.e. for all t and all $k \in \{-1, 0, 1\}$, the weights used in the convolution from $H(t, l)$ to $H(t + 1, l + k)$ are identical across time steps.

2.2. Output in the bottom layer

In contrast to common CNN models, the output of our network is always obtained in the lowest layer of the network. This structural property allows us to produce detailed outputs at input resolution. We add a time delay between input and output, ensuring that the inputs from the last presented frame were able to reach the topmost layer and return to the bottom layer before evaluating the loss.

A final convolution without temporal weight sharing is used to extract one map per target class from the bottom layer. The cross-entropy loss is computed perpixel over all output maps.

2.3. Recurrent convolutional neural networks

We process the images of videos sequentially, one image per time step. The state (i.e., the activations) at time t containing information about its past is combined with the image at time t, producing an output and a new state. Since the last output benefits from learning from the whole sequence, it is natural to place the frame that we want to evaluate at the end.

The first temporal copy is special, since it contains regular feed-forward connections. This allows us to produce activations in each layer such that all connection types can be used in the transition from t to $t + 1$.

Network depth. When processing input at time t, we allow $L - 1$ time steps for the information to reach the top level of the network and the same amount for propagating back to the bottom layer, where the output corresponding to time t is evaluated. Note that the last temporal steps do not need all the hidden layers, since their activation would no longer propagate to the output.

Our RNN is trained with backpropagation through time (BPTT), and can be interpreted as a very deep non-recurrent net after unfolding in time. In this non-recurrent network, multiple paths lead to the output, with the shortest path – from input $t = T$ to the final output – having only length $2L - 1$, and the longest $2L + t$, which amounts to a depth of 14 layers for our $L = 3, T = 8$ network.

Weight initialization and optimization. We initialize the weights and biases from a Gaussian distribution. It is important to ensure that the activations do not explode or vanish early during training. Ideally, activations in the first forward pass should have similar magnitudes. This is difficult to control, however. Instead, we choose the standard deviation of the weights for each layer l according to the scheme proposed by He, Zhang, Ren, and Sun (2015),

$$\sigma = \sqrt{\frac{2}{{k_l}^2 \cdot d_{l-1}}}, \tag{1}$$

which takes into account the filter size k_l and the number of filters of the last layer d_{l-1}. We determine the mean of the bias such that the average of the activations in every point of our network is positive and slightly decreasing over time. Liang and Hu (2015) use local contrast normalization at all layers to the same effect, which requires more GPU memory for the hidden layer activations. Due to our larger inputs and outputs and the increased number of time steps, current GPU memory restrictions prevent us from doing the same.

We learn the parameters of our network with backpropagation through time (BPTT) using RMSProp, which is a variant of resilient backpropagation (RPROP, Riedmiller & Braun, 1993) suitable for mini-batch learning (Dauphin, de Vries, Chung, & Bengio, 2015). RPROP and RMSProp to a large degree consider only the sign of the gradient, thus being robust against vanishing and exploding gradients, both common phenomena in RNN training.

During learning, we apply dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Combining dropout with RNNs is delicate, however. If it affects recurrent connections, their ability to learn long-range dependencies suffers (Pham, Bluche, Kermorvant, & Louradour, 2014). Thus, we apply dropout only to the final convolution with non-shared weights that computes the network output, and we find that using dropout consistently improves our results.

3. Related work

Multi-class semantic segmentation of images has a long history. Early approaches (e.g. He, Zemel, & Carreira-Perpiñán, 2004; Shotton, Winn, Rother, & Criminisi, 2006) optimize continuity over pixels in a conditional random field (CRF) with pairwise affinity potentials and unary potentials from a learned classifier. Since pixels contain only a limited amount of information, classifiers take regions into account. Most works (e.g. Batra, Sukthankar, & Chen, 2008; Russell, Kohli, & Torr, 2009) start with an unsupervised method to oversegment images into homogeneous regions called superpixels. The neighborhood relations of superpixels define a graph, whose node labels can be optimized in a CRF. More recently, approaches using neural networks have been successful (Ciresan, Giusti, Gambardella, & Schmidhuber, 2012; Schulz & Behnke, 2012; Sermanet et al., 2014). These approaches can model classification and smoothness simultaneously, but – in line with our findings here – can still profit from CRF postprocessing (Chen, Papandreou, Kokkinos, Murphy, & Yuille, 2014, 2015) and oversegmentation (Gupta, Girshick, Arbeláez, & Malik, 2014). Liu, Li, Luo, Loy, and Tang (2015) and Zheng et al. (2015) even suggest choosing the last part of the neural network architecture in a way that it resembles trainable versions of these postprocessing mechanisms. Long, Shelhamer, and Darrell (2015a) then demonstrated that a high resolution output can also be obtained without postprocessing using fully convolutional networks, which are created by replicating ImageNet-trained image classification networks over the image and finetuning them on the segmentation task. Noh, Hong, and Han (2015) further improve high resolution object delineation by refining and classifying object proposals with deconvolutional networks. While these approaches yield high accuracy on the PASCAL VOC segmentation challenge, they rely heavily on pre-training on a separate large classification dataset.

Multiple works address the integration of the depth component in RGB-D image labeling, especially with the advent of structured light sensors. Most notably, Shotton et al. (2013) used a random forest and depth difference features for efficient segmentation of depth images. Müller and Behnke (2014) extended the superpixel and CRF approach to RGB-D. Couprie, Farabet, Najman, and LeCun (2013) found that locally normalized depth and averaging in temporal superpixels improves CNN segmentation. Most recent work (e.g. Eigen & Fergus, 2015; Long, Shelhamer, & Darrell, 2015b) uses networks pre-trained on a large-scale classification task, which generally improves performance (Sharif Razavian, Azizpour, Sullivan, & Carlsson, 2014).

Several groups have used neural networks to process image sequences. Most works use stacks of frames to provide temporal context to the neural network. Le, Zou, Yeung, and Ng (2011) and Taylor, Fergus, LeCun, and Bregler (2010) learn hierarchical spatio-temporal features on video sequences using Gated Convolutional Restricted Boltzmann Machines (convGRBM) and Independent Subspace Analysis (ISA), respectively. Their image features are not learned discriminatively and the models do not allow localized predictions.

Simonyan and Zisserman (2014) use a two-stream architecture for action recognition, which separately creates high-level features from single-frame content and motion. Motion is provided through a stack of optical flow images, so that the modeled complexity is limited by the stack size. In our experiments, we found that increasing temporal context by providing frames consecutively yields improved performance.

More recently, Michalski, Memisevic, and Konda (2014) introduced a model designed to explicitly represent and predict the movement of entities in images. The architecture is chosen in a way that higher layers represent invariances of derivatives of positions (motions, accelerations, jerk). Our models do not explicitly model motion. However, our models can make use of deep layers even in the case no high-level position invariances exist, since in addition to motion, they also encode static content. Furthermore, in our model, deep layers have a lower resolution and facilitate transport of information across longer distances.

Jung, Hwang, and Tani (2014) introduce a multiple timescale recurrent neural network for action recognition, which uses neurons with fixed time constants. The model uses leaky integrator neurons, which limits the rate at which higher layer activations can change. It is trained and evaluated on a simplified version of the Weizmann Human Action Recognition dataset.

Various architectures for processing video data are explored by Karpathy et al. (2014). The architecture most similar to our model, *slow fusion*, uses weight sharing between time steps and merges them in higher layers. In their study, *slow fusion* yields best results. In contrast to classifying video sequences with a single label, we produce label output at pixel level.

RNNs were successfully used for object class segmentation by Graves (2012) and Pinheiro and Collobert (2014). Both works use recurrence only to increase spatial context, whereas we extend processing to the temporal domain.

Object recognition is another task where RNNs achieved state of the art results. In a recent work, Liang and Hu (2015) use convolutional layers unfolded in time similar to ours. Their architecture consists of a stack of five such layers interleaved with pooling and dropout layers. Similarly to previous works, Liang and Hu (2015) only use static information. Nevertheless, their model obtained superior results classifying images of multiple datasets.

Long-Short Term Memory (LSTM) units are capable of carrying information, at the original resolution, over long temporal distances. LSTM is often used in speech recognition (Graves, Abdelrahman, & Hinton, 2013) or language understanding (Sundermeyer, Schlüter, & Ney, 2012), where e.g. a specific property of a distant word or sound might influence the semantics of the current input. In this paper, we opt for simple neural units instead. While we are also interested in learning long-range dependencies, we do not provide spatial or temporal context at the original resolution. Instead, our architecture maintains expressive low resolution context information in higher layers. This is more realistic for natural images, where correlations are stronger between nearby pixels than those between distant ones. It also allows for sparser connectivity between units, since temporally distant units do not need to be wired.

Recent work of Bogun, Angelova, and Jaitly (2015) uses LSTM units for object recognition in video sequences. They obtained state of the art results on the Washington Dataset (Lai, Bo, Ren, & Fox, 2011) by incorporating information from several frames. Their most successful strategy was to train the network unidirectionally and to use a bidirectional model, based on the same set of weights, during prediction.

Sohn, Yan, and Lee (2015) introduce the Conditional Variational Autoencoder (CVAE). A CVAE is a conditional image label distribution $p(y|\hat{y}, z)$, where x is the input image, $\hat{y}$ is an initial guess of the output, and $z \propto p(z|x, \hat{y})$ is a latent variable. It is possible to sample from the CVAE and extend it to multiple scales. Our hierarchy of deterministic hidden states is also conditioned on the input and modulates a high resolution image label prediction. However, our proposed model employs recurrent connections not only to refine an initial guess in a single time step, but also to incorporate new knowledge from multiple consecutive video frames.

Pascanu, Mikolov, and Bengio (2013) suggest that LSTM also addresses the problem of vanishing gradients. Here, we use RMSProp as optimization method, which – in addition to preventing vanishing gradients – also counteracts gradient explosion.

Our architecture choices are motivated by the Neural Abstraction Pyramid of Behnke (2003), which performs pixel-wise classification tasks at input resolution as well. In contrast to our work, Behnke did not train on video sequences, but only on stationary patterns, which in some cases were corrupted by temporally changing noise. We also include modern architectural features, such as max-pooling and ReLU non-linearities, dropout, and use RMSProp to increase learning speed.

4. Experiments

We first conduct experiments on handcrafted datasets, which allow us to demonstrate important characteristics of our model. In a second step, we use our architecture for object class segmentation on a challenging RGB-D dataset.

4.1. Toy experiments

We present three toy experiments, showing that our network is able to learn (1) filtering noisy information over time, (2) tracking and interpreting motion, and (3) retaining an internal state including uncertainty.

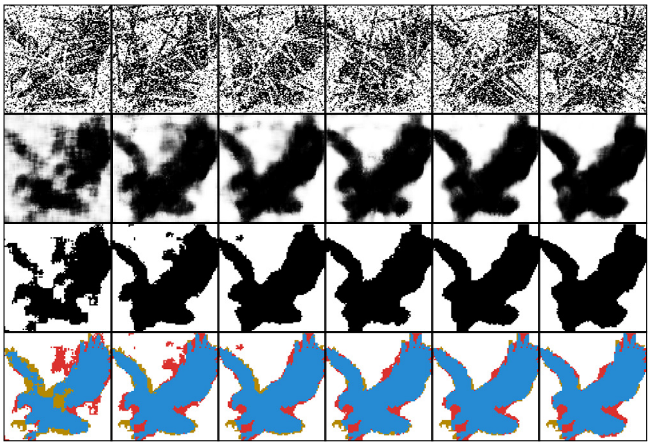


Fig. 3. Toy experiment: Denoising. Rows represent, in order: the RGB input of the network for each time-step, the output of the softmax layer, the final outputs of the network, the evaluation (● True Positives ○ True Negatives ● False Positives ● False Negatives). The last output is used for evaluation of pixelwise classification error.

Table 1
Denoising results for different models.

Method	Accuracy
RNN	**93.3**
CNN (all timesteps)	89.2
Thresholded average	81.6

4.1.1. Denoising

In this experiment, we feed different degraded versions of the same binary image to the network. We use salt and pepper noise, uniformly distributed over the whole image. We also draw random black or white lines, to make the task more difficult. The task is to obtain the original image without noise. One way the network could solve this task would be to learn to average the image over time (which is our baseline). In addition, denoising filters learned by the neural network can remove high frequency noise.

To ensure that the network is able to generalize instead of learning an input by heart, we use different objects for training, validation and testing. Every split contains 100 independently generated sequences.

Since the task has a low complexity, we opt for a simple convolutional model of only one hidden layer with 32 maps. A small filter size of 5×5 provides sufficient spacial context. We use $T = 6$ temporal copies. During training, we optimize a weighted sum of the losses at all time steps, with a ten times larger weight placed on the final output. In all toy examples, we train for 12 000 iterations with minibatches of size 16. For quantitative comparisons, we use the average class accuracy, i.e., the average of the diagonal of the confusion matrix between foreground and background pixels, after the confusion matrix rows have been normalized with the class prior.

Fig. 3 shows an example from the test set. Our model is able to improve its prediction step by step, accumulating over time information even from the areas which are more affected by noise. After only two steps, the network is able to remove most of the false positives and to assemble together almost all features of the object. Table 1 and Fig. 4 show that the RNN performance compares favorably to a naïve thresholding of the average image.

We also train a baseline convolutional model without recurrent connections, but having access to the same amount of data. For this purpose, we concatenate all images of the sequence and present them to the network simultaneously. The predictions produced by

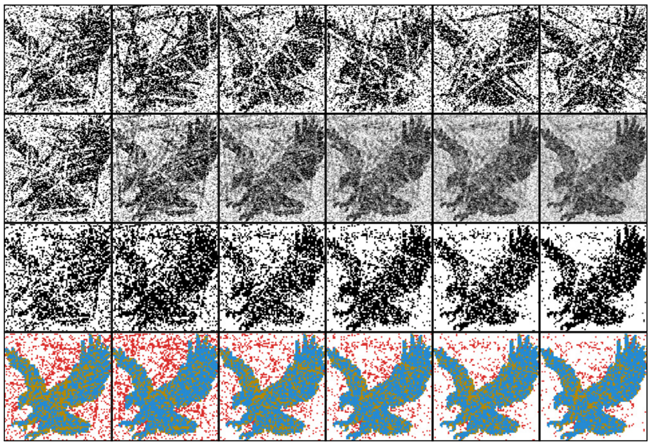


Fig. 4. Toy experiment: Moving average of the inputs presented to the network. Rows represent, in order: the RGB input of the network for each time-step, the moving average of the inputs, the final outputs after thresholding, the evaluation (● True Positives ○ True Negatives ● False Positives ● False Negatives). The last output is used for evaluation of pixelwise classification error.

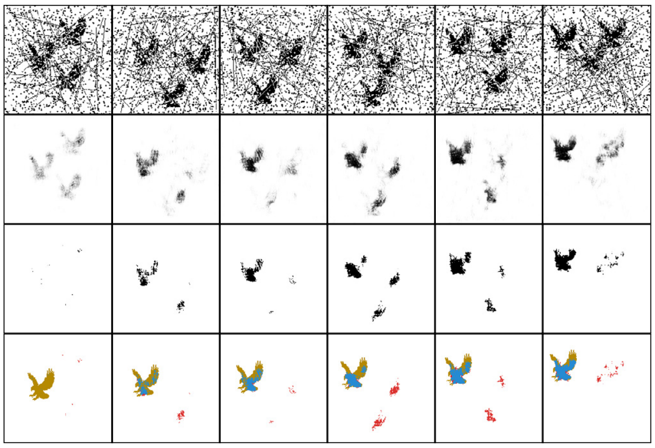


Fig. 5. Toy Experiment: Detecting movement. Rows represent, in order: the RGB input of the network for each time-step, the output of the softmax layer, the final outputs of the network and the evaluation (● True Positives ○ True Negatives ● False Positives ● False Negatives). The rightmost output is used to determine classification error.

this baseline model (images not shown) indicate that it learned to combine information from multiple frames. However, the baseline accuracy of 89.2% is inferior to the accuracy of the recurrent architecture (93.2%).

4.1.2. *Detecting movement*

In this experiment, we test the capabilities of the network to track a foreground object moving with constant speed through a noisy image. We use the same task as in the previous section (denoising a binary image), however, but this time we choose a smaller black object which is moving on a white canvas. Furthermore, to ensure that motion is the cue for tracking, we add two randomly placed distractor objects of the same shape and size in a random position at every time step. These distractors should be classified as background. To prevent the network from overfitting on motion direction and speed, we generate several sequences, each moving the object from a random position to another, with varying speed. The image size was 200×200, object at most 50 px in their largest dimension, and speed was between 9 and 23 pixels per time step.

Fig. 5 shows the results obtained on this task using the unidirectional network. In the first time step, the network cannot decide which object is moving continuously. Already at $t = 2$, however, the network detects a slight positional change from one of the objects, while the others are further away from their initial position. The softmax layer activations (second row) show that the certainty of the hypothesis increases step by step. Also, one can notice that more details are added to the representation. Some false positives still exist when the new random position of a distractor object is close to its former position.

The baseline that we use for this experiment is also a convolutional neural network receiving the whole sequence input at once. We used a three layer architecture, as in the case of the recurrent counterpart. Our results show significant differences between the two models with respect to class accuracy. The recurrent network achieves a 89% accuracy, compared to 76.1% for the baseline.

4.1.3. *Retaining uncertainty*

While in the previous experiments, we showed that the network is able to track a moving object, we now consider if a regular movement can be inferred from temporally distant information. We construct a dataset where two input frames are provided, with the same object present in different locations. Assuming linear motion, we want to predict the position of the object in the center. This task would be easy if both input frames were known at the same time, but we restrict access in our model as follows.

We use a bi-directional version of our model with weights shared between the past and future network parts. The two input images are provided to the first timestep of the "past" network and the first timestep of the "future" network, respectively. In this setup, the initial positions have to be remembered until information from the past and future converges at the center time step. Since denoising is not an essential component of this task, we do not add noise.

Fig. 6 depicts a sample sequence from the test set. As no motion information is provided, the best strategy of the network is to create a circular expanding hypothesis from the seen location. This time-dependent distribution over possible locations collapses when hypotheses from both timelines are combined. This is what we observe in the output maps of Fig. 6. While the position is correctly identified, the shape of the object is largely lost.

As a baseline model, we also use a CNN with access to the same amount of data as our RNN. Due to the large spatial distance between objects, the baseline model is unable to determine the intermediate position in the motion. This leads to a class accuracy of merely 51.3%. Increasing the size of the filters from 7 to 13 leads to a class accuracy of 64.3%. Even with such an advantage, the baseline is more than 10% worse than the RNN, which achieves a class accuracy of 74.9% for this task.

4.2. *RGB-D object class segmentation*

The NYU-Depth v2 (NYUD, Silberman, Hoiem, Kohli, & Fergus, 2012) dataset is comprised of video sequences taken from 464 indoor scenes, annotated with a total of 894 categories. We use the popular relabeling into four high-level semantic categories, small objects that can be easily carried ("prop"), large objects that cannot be easily carried ("furniture"), non-floor parts of the room: walls, ceiling, columns ("structure"), and the floor of the room ("ground").

Although NYUD was recorded as a video sequence, a subset of 1449 frames which were preprocessed and manually labeled is prevalent in the literature. The remainder – 407 024 frames – consists of raw RGB-D camera images.

To transform the dataset into an image sequence dataset, but at the same time use the labeled frames for evaluation, we

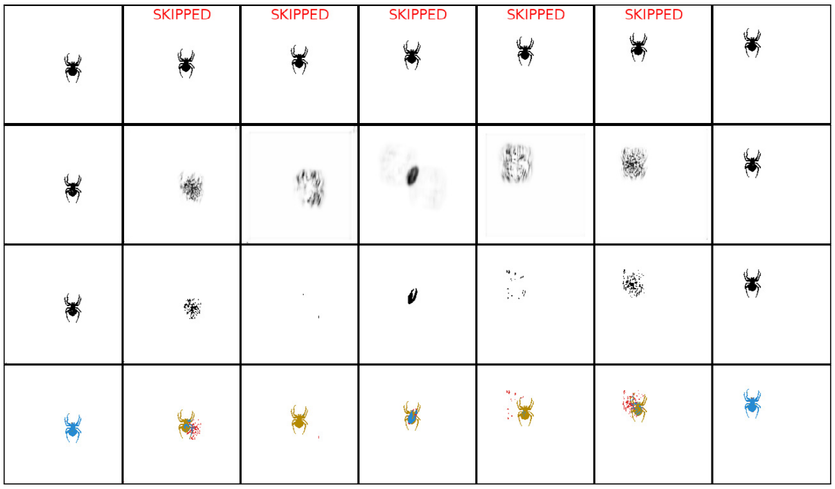


Fig. 6. Toy experiment: Retaining uncertainty. Rows represent, in order: the RGB input of the network for each time-step, the output of the softmax layer, the final outputs of the network and the evaluation (🔵 True Positives ○ True Negatives 🔴 False Positives 🟤 False Negatives). Center output of the bidirectional network is used to determine classification error.

extract the past and future context of each labeled frame from the video stream and preprocess it similar to the labeled frames. For evaluation, we compare outputs corresponding to the labeled frame with the ground truth, retaining the same training/testing split as in the literature.

To preprocess the RGB and depth images, we follow standard procedure, sequentially applying lens correction,[1] projecting the depth readings into the RGB sensor frame, and filling-in missing depth readings (Levin, Lischinski, & Weiss, 2004).

To obtain ground truth information for unlabeled frames in our sequences, we propagate labels along optical flow directions. Due to manual labeling of the dataset, classes are often separated by small unlabeled regions, especially at edges which are crucial to the optical flow estimation. To address this problem, we use a colorization method (Levin et al., 2004) to determine missing labels close to labeled regions. Only the training set is modified, which allows direct comparison with other methods on the test set. After label fill-in, we compute optical flow (Farnebäck, 2003) on RGB image pairs. When label information is unavailable or ambiguous due to limitations in the optical flow computation, we write *unknown* labels, which are excluded from the computation of the loss and its gradient.

We also use optical flow to select which images are included in a sequence. Fast displacements are tricky to detect (Brox & Malik, 2011), while too slow motion results in quasi-static images, foiling our attempt to exploit the temporal context. Since the dataset was recorded at 30 Hz, taking every frame would result in no visible motion at input resolution. We thus decide to add a new image to the sequence once the optical flow shows, on average, a motion larger than half of our filter size.

Not all sequences in the dataset contain sufficient temporal context. In such cases, we complete them by replicating the first or last available frame.

4.2.1. *Learning*

We train the network with depth $L = 3$, an input resolution of 160×160 pixels, again using mini-batches of size 16, and a temporal context of 8 frames. As input, we use Histogram of Oriented Gradients (HOG) and Histogram of Oriented Depth (HOD) channels, the whitened version of the images (cf. Höft, Schulz, & Behnke, 2014), an estimated height map and the optical flow.

[1] Computed by OpenCV's "Camera Calibration and 3D Reconstruction" package.

The height map was computed by (i) clustering normals in the depth map, (ii) projecting onto the most vertical cluster mean, determining the lowest point, and (iii) calculating the relative height of all other points in the depth map.

We use randomly chosen 10% of the training set for validation (early stopping and model selection). While images are randomly transformed during training, we use a fixed, randomly picked set of transformations for validation to ensure stable estimates. The loss is measured at time steps $t = 3$, 6, and 8. Training continues for 12 000 iterations, with an initial learning rate of $3 \cdot 10^{-4}$. The learning rate was automatically decreased three times when the validation error failed to improve. Once learning stopped, we retained the model which obtained the best performance on the validation set.

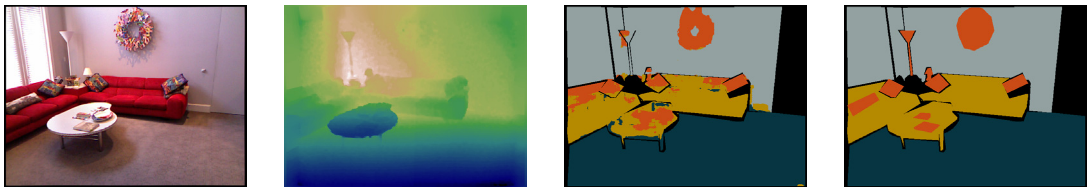
Fig. 8 shows the evolution of the prediction for the *structure* class over the 8 input frames. The first row shows the RGB part of the input image, the second the softmax output for *structure*. The third row shows for which pixels the predicted class is *structure*, and the fourth row compares this prediction to ground truth when applicable (time steps 3, 6 and 8). As in Fig. 7, we overlay "unknown" labels from ground truth to aid orientation in the image. Over time, the prediction is refined and false positives are reduced. It also seems that the network has learned to treat time steps 3 and 6, where the intermediate loss is measured, differently than other time steps.

4.2.2. *Depth-normalization*

We integrate depth-normalized covering windows (Schulz, Höft, & Behnke, 2015a) into our learning algorithm. This approach evaluates the model on image patches at a spatial resolution which is dependent on the distance of the center pixel to the observer, effectively building scale invariance and adaptive output resolution into the model. Similar to Schulz et al. (2015a), this patch-based approach allows us to use smaller maps, which speeds up the training dramatically (days vs. weeks). As input, we used 80×80 instead of 160×160 pixels, and scaled hidden layers accordingly. At prediction time, the network has to process patches that fully cover the original image and combine their prediction afterwards. Thus, speed depends on the number of patches which in turn depends on the depth distribution of the RGB-D image.

4.2.3. *Comparison with state of the art*

Table 2 shows our result with and without depth-normalized covering windows (CW) together with state-of-the-art results on



(a) RGB frame. (b) Depth. (c) Prediction. (d) Ground truth.

Fig. 7. Prediction for one of the NYUD dataset frames. Images (a) and (b) show RGB and depth, respectively, after being preprocessed. (c) and (d) represent the prediction and ground truth, respectively, where color codes "floor" (●), "prop" (●), "furniture" (●), "structure" (●) and "unknown" (●). The network detects most of the pixels correctly, even some wrongly labeled ones (e.g. the third object on the table and the center of the wall-mounted piece). Note that in (c), "unknown" is superimposed from ground truth to aid comparison with ground truth. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Comparison of NYUD classification performance to other state-of-the-art approaches without transfer learning. Our model with enabled covering windows (CW), unsupervised weight initialization (WI), and conditional random fields (CRF). Input consists of color/depth information, optionally with height (H). Baselines use convolutional neural networks (CNN), random forests (RF), and self-localization and mapping (SLAM). The "all-frames" model receives inputs from all frames at every time step.

	Class accuracies (%)				Average (%)	
Method	Ground	Struct	Furnit	Prop	Class	Pixel
Ours (CW, RGB-D only)	78.6	49.2	48.7	48.3	56.2	52.0
Ours (CW)	95.8	74.6	54.2	**64.0**	72.1	68.6
Ours (WI+CW)	94.9	76.8	65.5	60.8	74.5	73.1
Ours (WI)	94.3	83.7	72.0	54.9	76.2	76.4
Ours (WI+CW+CRF)	95.4	78.9	67.3	60.8	75.6	74.6
Ours (WI+CRF)	94.2	83.9	**72.0**	56.3	**76.6**	**77.2**
All-frames	**97.2**	70.0	51.1	56.0	68.6	64.6
Schulz et al. (2015a) (CNN + CRF)	93.6	80.2	66.4	54.9	73.7	73.4
Müller and Behnke (2014) (RF + CRF)	94.9	78.9	**71.1**	42.7	71.9	72.3
Stückler et al. (2013) (RF + SLAM)	90.8	81.6	67.9	19.9	65.0	68.3
Couprie et al. (2013) (CNN)	87.3	**86.1**	45.3	35.5	63.5	64.5
Silberman et al. (2012) (RF)	68.0	59.0	70.0	42.0	59.6	58.6

the same dataset. We compare against other methods which do not use transfer learning (e.g. by initializing filters using ImageNet-based transfer learning). Our RNN (WI+CW) outperforms all other published approaches. Overall, depth-normalization (CW) turns out to be worse for RNN, maybe indicating that there is not enough context to be interpreted in the chosen window sizes. However, CW is much better at identifying small objects (*prop*), at up to 64% vs. 54.9% for full frames.

Note that when using RGB-D input only, overall class accuracy drops to 56.2%. With larger datasets, better RNN pretraining and growing GPU memory, larger models could be learned, removing the strong dependence on preprocessing.

We also investigated whether presenting video frames sequentially is an effective strategy. We used the best model and hyper-parameters, but provided all eight inputs at every RNN time step. This approach, labeled "all frames", achieves significantly lower accuracy than the sequential model, despite the increased computational effort.

The outputs of our RNN can still be improved through other means. Instead of determining a pixel-wise maximum, we supply the probabilities as data term to the conditional random field (CRF) of Müller and Behnke (2014). The result (WI+CW+CRF and WI+CRF) improves performance even further. This indicates that our predictions are better suited for CRF postprocessing than CNN (Couprie et al., 2013; Schulz et al., 2015a) and random forests (Müller & Behnke, 2014; Schulz, Waldvogel, Sheikh, & Behnke, 2015b; Silberman et al., 2012).

Comparison to transfer learning. Similar to non-recurrent neural networks (e.g. Erhan et al., 2010), our RNN profits from a weight initialization (CW vs. WI+CW). Supervised weight initialization using transfer learning on the large ImageNet database led to the (to the best of our knowledge) current top result on NYUD by Eigen and Fergus (2015), with a class accuracy of 82.0%. Our RNN only narrows the gap to the transfer learning approach. Weight initialization for recurrent connections is, therefore, an open but promising research direction.

4.2.4. Does temporal context help?

Except for Stückler, Waldvogel, Schulz, and Behnke (2013), none of the publications in Table 2 made use of temporal context to determine class labels. We would like to know whether our improvement is due to the fact that we use additional training data (albeit without labels), or through recurrent processing.

To check whether our network takes advantage of temporal context, we perform a static frame experiment. We use the same frame as input at all time steps during training and prediction. In this setting, the recurrent architecture is still able to learn long-range spatial dependencies, but cannot exploit temporal context, which results in accuracy reduction in both class accuracy and pixel-wise accuracy (2.7 and 5.5 percentage points, respectively) relative to the model which has access to temporal context. This model still outperforms non-recurrent models, showing that spatial recurrent processing is the essential improvement.

When using static frames as above, we remove both, temporal context and additional ground truth generated through optical flow. Removing only additional ground truth during training results in a reduction of the class accuracy by 2%, while the pixel-wise accuracy decreases by 3.1%. These controls suggest that the superior accuracy of the RNN is mainly due to recurrent spatial and temporal processing, and intermediate ground truth is important to help this very deep – and mostly uninitialized – network learn.

5. Conclusion

In this work, we introduced a recurrent convolutional neural network architecture, which in addition to learning spatial

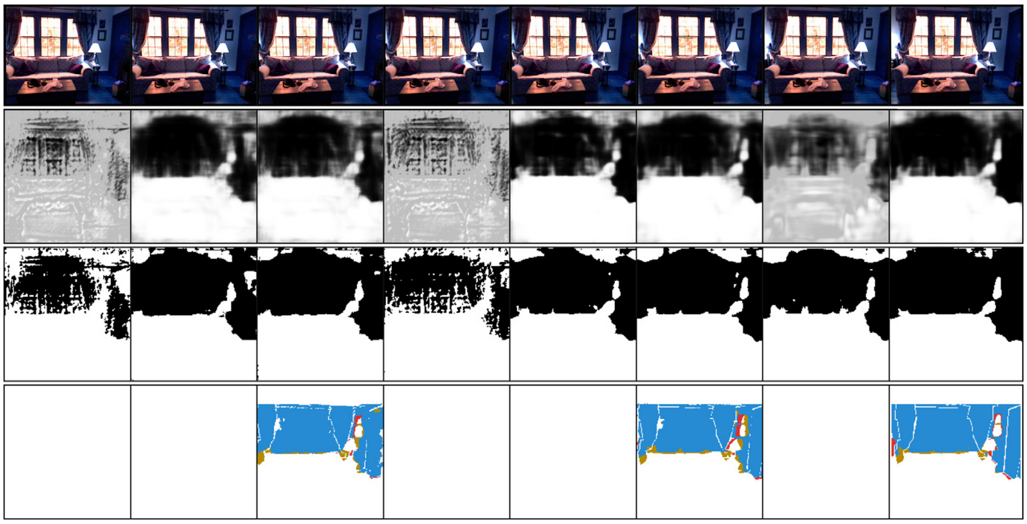


Fig. 8. Prediction for class *structure* on a sample of the NYUD test set. Rows represent, from top to bottom: the RGB input, the softmax layer output, the output of the network; and the evaluation (● True Positives ○ True Negatives ● False Positives ● False Negatives). Note that the loss is measured only at time steps 3, 6 and 8, with a reduced loss weight of 0.1 at time steps 3 and 6.

relations is also able to exploit temporal relations from video. We started with a series of toy examples that showed that our networks are able to solve tasks that require denoising, detecting movement, and retaining uncertainty.

We further carried out experiments on sequences of indoor RGB-D video sequences from the NYUD dataset. Combined with dropout, unsupervised weight initialization, covering windows and conditional random fields, our proposed model improves performance when compared to other non-recurrent baseline models and random forests, obtaining the best results obtained so far without transfer of features from ImageNet.

References

Batra, D., Sukthankar, R., & Chen, T. (2008). Learning class-specific affinities for image labelling. In *Conference on computer vision and pattern recognition, CVPR*.

Behnke, S. (2003). *Lecture notes in computer science*: *Vol. 2766. Hierarchical neural networks for image interpretation*. Springer.

Bogun, I., Angelova, A., & Jaitly, N. (2015). Object recognition from short videos for robotic perception. arXiv preprint arXiv:1509.01602.

Brox, T., & Malik, J. (2011). Large displacement optical flow: descriptor matching in variational motion estimation. *Transactions on Pattern Analysis and Machine Intelligence*, *33*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A.L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2015). Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *International conference on learning representations, ICLR*.

Ciresan, D., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems, NIPS*.

Couprie, C., Farabet, C., Najman, L., & LeCun, Y. (2013). Indoor semantic segmentation using depth information. In *International conference on learning representations, ICLR*.

Dauphin, Y.N., de Vries, H., Chung, J., & Bengio, Y. (2015). Rmsprop and equilibrated adaptive learning rates for non-convex optimization. arXiv preprint arXiv:1502.04390.

Eigen, D., & Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *International conference on computer vision, ICCV*.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research (JMLR)*, *11*, 625–660.

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In *Image analysis*. Springer.

Graves, A. (2012). *Studies in computational intelligence*: *Vol. 385. Supervised sequence labelling with recurrent neural networks*. Springer.

Graves, A., Abdelrahman, M., & Hinton, G. E. (2013). Speech recognition with deep recurrent neural networks. In *International conference on acoustics, speech and signal processing, ICASSP*.

Gupta, S., Girshick, R., Arbeláez, P., & Malik, J. (2014). Learning rich features from rgb-d images for object detection and segmentation. In *European conference on computer vision, ECCV*.

He, X., Zemel, R. S., & Carreira-Perpiñán, M. Á. (2004). Multiscale conditional random fields for image labeling. In *Conference on computer vision and pattern recognition, CVPR*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. arXiv preprint arXiv:1502.01852.

Höft, N., Schulz, H., & Behnke, S. (2014). Fast semantic segmentation of RGB-D scenes with GPU-accelerated deep neural networks. In *German conference on artificial intelligence, KI*.

Jung, M., Hwang, J., & Tani, J. (2014). Multiple spatio-temporal scales neural network for contextual visual recognition of human actions. In *International conference on development and learning and on epigenetic robotics, ICDL*.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Conference on computer vision and pattern recognition, CVPR*.

Konda, K.R., Memisevic, R., & Michalski, V. (2013). Learning to encode motion using spatio-temporal synchrony. arXiv preprint arXiv:1306.3162.

Lai, K., Bo, L., Ren, X., & Fox, D. (2011). A large-scale hierarchical multi-view rgb-d object dataset. In *International conference on robotics and automation, ICRA*.

Le, Q. V., Zou, W. Y., Yeung, S. Y., & Ng, A. Y. (2011). Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Conference on computer vision and pattern recognition, CVPR*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Levin, A., Lischinski, D., & Weiss, Y. (2004). Colorization using optimization. In *Special interest group on graphics and interactive techniques, SIGGRAPH*.

Liang, M., & Hu, X. (2015). Recurrent convolutional neural network for object recognition. In *Conference on Computer Vision and Pattern Recognition, CVPR*, June.

Liu, Z., Li, X., Luo, P., Loy, C. C., & Tang, X. (2015). Semantic image segmentation via deep parsing network. In *International conference on computer vision, ICCV*.

Long, J., Shelhamer, E., & Darrell, T. (2015a). Fully convolutional networks for semantic segmentation. In *Conference on computer vision and pattern recognition, CVPR*.

Long, J., Shelhamer, E., & Darrell, T. (2015b). Fully convolutional networks for semantic segmentation. In *Conference on computer vision and pattern recognition, CVPR*.

Michalski, V., Memisevic, R., & Konda, K. (2014). Modeling deep temporal dependencies with recurrent grammar cells. In *Advances in neural information processing systems, NIPS*.

Müller, A.C., & Behnke, S. (2014). Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images. In *International conference on robotics and automation, ICRA*.

Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. In *International conference on computer vision, ICCV*.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *Journal of Machine Learning Research (JMLR)*, *28*, 1310–1318.

Pham, V., Bluche, T., Kermorvant, C., & Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. In *International conference on frontiers in handwriting recognition, ICFHR*.

Pinheiro, P. H., & Collobert, R. (2014). Recurrent convolutional neural networks for scene labeling. In *International conference on machine learning, ICML*.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *International conference on neural networks*.

Russell, C., Kohli, P., & Torr, P. H. et al. (2009). Associative hierarchical crfs for object class image segmentation. In *International conference on computer vision, ICCV*.

Schulz, H., & Behnke, S. (2012). Learning object-class segmentation with convolutional neural networks. In *European symposium on artificial neural networks, ESANN*.

Schulz, H., Höft, N., & Behnke, S. (2015a). Depth and height aware semantic RGB-D perception with convolutional neural networks. In *European symposium on artificial neural networks, ESANN*.

Schulz, H., Waldvogel, B., Sheikh, R., & Behnke, S. (2015b). CURFIL: Random forests for image labeling on GPU. In *International conference on computer vision theory and applications, VISAPP*.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International conference on learning representations, ICLR*.

Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer vision and pattern recognition (CVPR) workshops*.

Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., et al. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*,.

Shotton, J., Winn, J., Rother, C., & Criminisi, A. (2006). Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European conference on computer vision, ECCV*.

Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In *European conference on computer vision, ECCV*.

Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems, NIPS*.

Sohn, K., Yan, X., & Lee, H. (2015). Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems, NIPS*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, *15*, 1929–1958.

Stückler, J., Waldvogel, B., Schulz, H., & Behnke, S. (2013). Dense real-time mapping of object-class semantics from RGB-D video. *Journal of Real-Time Image Processing*,.

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). LSTM neural networks for language modeling. In *Interspeech*.

Taylor, G. W., Fergus, R., LeCun, Y., & Bregler, C. (2010). Convolutional learning of spatio-temporal features. In *European conference on computer vision*, (ECCV). Springer.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., & Du, D. et al. (2015). Conditional random fields as recurrent neural networks. In *International conference on computer vision, ICCV*.